A Meta-Model for Ontologies with ORM2

Christina Tziviskou[1] and C. Maria Keet[2]
[1] Politecnico di Milano, via Ponzio 34/5, 20124 Milano, Italy
[2] Faculty of Computer Science, Free University of Bozen-Bolzano, Italy
tzivisko@elet.polimi.it, keet@inf.unibz.it

Abstract. Development of ontology development tools and ontology-enhanced software applications requires thorough understanding of ontology languages in order to implement them according to their specification. We present a formal specification of the ontologies part of the Web Services Modeling Language WSML documentation as a conceptual model in ORM2. Such an approach abstracts the semantics about ontological constructs, axioms, and properties from their implementation in arbitrary formats, thereby making the model easily understandable and reusable. This formal model in ORM2, which is understandable by both logician and software developer, can be used as any other conceptual model to develop applications, thereby ensuring smooth transition from theory to implementations that are faithful to the theory.

Keywords. WSML, ontologies, ORM2, meta-modeling

1 Introduction

During the last decade, the increasing use of ontologies in software and Web engineering has also increased the number of languages used for expressing them. Therefore, the choice of the appropriate ontology language becomes an important issue, in particular for the development of ontology development tools and ontology-enhanced information systems. These tasks are not straightforward, largely because they depend on the understandability of the language specification by software developers. To assist comprehension by engineers, and usability and consistent reuse of the ontology languages across software applications, the need arises to have a conceptual model of the ontology language. With such a formal conceptual model—a meta-model of the ontology language—one can capture the ontology language specification explicitly, thereby resolving any ambiguities that may exist in its documentation. Such a formal conceptual model of the language and specification document can then be used in an automatic or semi-automatic way within applications for the development of ontologies and ontology-enhanced information systems; thus, facilitating the implementation of tools based on ontologies. However, no formal conceptual model of an ontology language exist yet that satisfies these needs.

The Web Services Modeling Language (WSML) [5, 21, 4] comprises five variants for modeling Web Services and ontologies. We focus on WSML ontologies represented with WSML-Flight, which is based on the Datalog subset of F-logic

extended (locally) with stratified negation, and the representation of the meta-model in Object-Role Modeling (ORM2) [9] in order to develop an ontology library. Our present aim is to extract the WSML ontology elements, analyze and explicitly describe them using the constructs of the ORM2 language. We have chosen ORM2, because it is the most expressive conceptual modeling language, thereby permitting representation of all explicit and implicit constraints of both the WSML language and its documentation; it has a formal underpinning, thereby enabling an unambiguous representation; and it is design and implementation independent so that the same ORM model may be used for both database development and other software applications, thereby ensuring interoperability among ontology tools.

The advantages of such an approach is that the analysis and formal specification is done once instead of separately by each developer, thereby reducing the risk of misinterpretations and errors by developers whose proficiency is software development as opposed to reading language specifications. The ORM2 model of WSML and its specification also serves as the communication bridge between logicians who developed WSML and software developers, thereby facilitating the development of ontologies valid with respect to the WSML language. This ORM2 specification can be expanded in order to express meta-models of other ontology languages. In this case, the conceptual model that we will present could become an effective engineering mechanism for comparing ontology languages and for expressing mappings between ontologies that were developed in different languages. In another direction, it may facilitate posing conceptual queries over the ontologies that are stored in a RDBMS repository generated from the conceptual model.

The structure of the article is as follows: in Section 2 we describe other conceptual modeling approaches for representing or using the meta-model of an ontology language; the main WSML and ORM elements are presented in Section 3; our approach of modeling WSML in ORM2 is presented in Section 4; in Section 5 we evaluate the model; and in Section 6 we draw some conclusions and future work.

2 Related Work

There are two principle distinct approaches to reuse database technologies for ontology engineering. First, reusing the conceptual modeling language or its diagrammatic representation as interface to the logical theory, and, second, storing ontologies in database, for which there is the need of a good conceptual data model as is the case for any database. In the current scope, the second approach is relevant: methodologies and techniques to store ontologies in databases. Large ontologies, such as the Gene Ontology [8], Foundational Model of Anatomy [16], and Cell Cycle Ontology [2], offer a “database version” of their ontologies, partly due to legacy issues and partly to enhance performance of querying over the ontologies. Their respective database schemas are poor in semantics with respect to the ontology language, such as storing only triples and annotation information, but not the full set of constraints available in an ontology language. In addition, there are ontology tools such as the instanceStore [3], QuOnto [1], and MASTRO [14], that store at least part of the ontology and its instances in a relational database to gain in performance. To

the best of our knowledge, no conceptual data model of their database is publicly available. For the widely used Protégé [15] ontology development tool, we could reverse engineer it. In Protégé, one can store an ontology in OWL format, RDF, frames (that end up in a relational database as persistent storage), or in a database specifically (MySQL). Although the Protégé interface gives the impression of a nicely structured information system, the underlying database is devoid of any semantics. The Protégé-database that stores the ontology consists of one long table, which, when reverse-engineered into a 'conceptual model', has meaningless value types such "short value". But it has also "facet", "frame", and "slot" as value types, thereby revealing the frame-based origin in a rudimentary version. This simplification is too poor to be considered a proper conceptual model. Furthermore, values stored in the table are often only numbers; hence, the intelligence of the tool is coded in the application layer and in doing so, the ontology in the database is not immediately reusable outside Protégé, let alone query the database directly [7]. Also, the probability of misinterpretations and errors is high since a user can extract at most the description of ontology elements but not the formal semantics. If one were to push the Protégé information system further to also use the database as a repository of *multiple* ontologies, then the system becomes unmanageable.

3 Background

WSML [6] is a language for modeling semantic Web Services. It provides the elements for expressing the Web services interfaces, capabilities, and interchanged data captured as ontologies. We focus on the part of WSML for modeling ontologies. These are defined in a frame-based style in terms of the conceptual syntax for WSML.

Any ontology is composed of *concepts* that describe elements of the domain. In the common case, concepts are related by is-a relationships building a hierarchy, and in more advanced schemas they are structured with common properties. The properties for a concept are defined internally with *attributes*, similar to the notion of attributes in UML and ER, and may have cardinality and relational constraints attached to them. In addition, there are *relations* between concepts defined externally, which are, according to WSML semantics, conceptually distinct from the attributes that link a base concept to other concepts. The role played by each concept participating in a relation is denoted by the appropriate *parameter*. Additional constructs for concepts refine the ontology scheme with intersection, union, and complement but are only defined in the logical expression syntax, and therefore can be used within rules in the logic attached to the ontology elements at the conceptual syntax through *axioms*. The elements of the domain instantiate the corresponding concepts and relations respecting the constraints imposed by their definitions, and are defined in WSML respectively as *instances* and *relation instances*. Furthermore, ontologies may be built in a modular way.

Recollecting briefly ORM [9] with respect to WSML, ORM diagrams are based on a few main constructs: object (or value) types (concepts in WSML) and fact types (relationships – either internal or external in WSML) composed of the roles that the object (or value) types play in the relation. Unlike WSML, ORM is an attribute-free

language; attributes in languages such as UML are modeled 'flat' as fact types. Further, the ORM language has powerful functional dependencies to constrain the roles played by object and value types. ORM allows capturing sophisticated relations among objects, largely motivated by ORM's implicit correspondence between data requirements expressed in natural language and facts modeled in ORM diagrams.

4 WSML in ORM2

This section presents the ORM2 diagrams that incorporate the WSML notation for ontologies. The diagrams have been developed in NORMA [12], a CASE tool for ORM2. Because of the powerful features of ORM2, the model presented here contains additional constraints that capture semantic aspects that are not explicitly defined in WSML or its documentation but limit human errors during ontology development.

4.1 Ontology

Each ontology (Fig. 1.) is identified by an International Resource Identifier (IRI). It is defined in exactly one document, and may contain a set of elements (defines fact type). Modularized ontology development in WSML is achieved by allowing ontologies to be imported and refined in the current one. In particular, imports is transitive; hence, the ontology imports also the definitions of the ontologies contained in the transitive closure of imports. Transitivity is a property that cannot be depicted by the graphical language of other modeling schemes (UML, ER). The model extends the WSML specification and explicitly indicates that no cycles are allowed in this relationship; thus, an ontology is not allowed to recursively import itself, thereby avoiding faulty modeling that can cause problems if it is propagated to a reasoning engine.

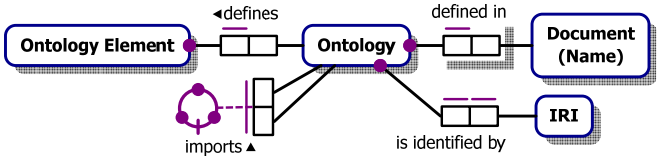

Fig. 1. Ontology entity type and its definition

4.2 Ontology Elements

An ontology describes elements of the domain and their phenomena. A simple way of describing a set of domain elements in WSML is by defining concepts that can be related to each other. In case the axioms that relate them are denoted as properties for the concepts, they are used to refine the concepts definition and exist internally as attribute definitions, otherwise they are defined globally in the ontology as relations.

Concepts and Instances. The main constructs for a concept are shown in Fig. 2. The Atomic entity type considers primitive concepts, which are not constructed on top of others but their definition can be refined by axioms presented in the following paragraph. The Negation, Disjunction and Conjunction capture the semantics of allowing complement, intersection and union of concepts. In WSML, the three last constructors exist at the logical expression syntax and not at the conceptual one.

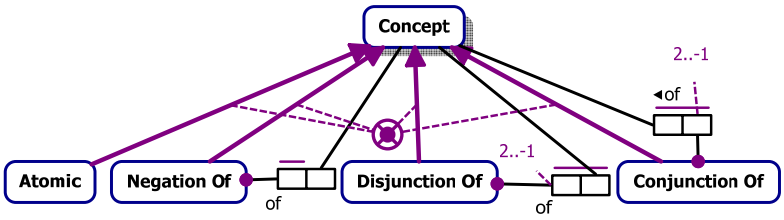

Fig. 2. Concept constructs

The relationships identified for a concept are: (i) ISA relationships, used to create hierarchy of concepts, commonly found in lightweight ontologies, (ii) relationships with other concepts, (iii) constraints on the relationships with concepts restricting the value space (range) for the related objects, and (iv) membership relationships indicating the objects instantiating the concept and/or fulfilling roles in the relationships for the concept.

In the diagram of Fig. 3, we depict the modeling of WSML concepts and instances with ORM2. The has super concept and is equivalent fact types denote hierarchy of concepts. The former is used to model the inclusion axiom that a concept is more specific than its super concepts; it is defined irreflexive and transitive, and therefore it inherits also the relationships defined in the more general ones. Constraints on these two fact types are denoted with ring constraints in ORM2. Cycles are permitted allowing equivalence of concepts to be deduced by reasoning engines, although a concept is not allowed to be indicated as more specific than itself because deducing equivalence in this case is redundant. The definition of is equivalent, is analogous and is used to model equality axioms. Semantic relationships with other concepts map to attribute definitions when they contribute to the concept definition. The fact types has concept type and has data type are mutually exclusive and denote the type of the related objects: a concept or a data type respectively. Multiple defined types are interpreted as their disjunction. The of constraining type reflects semantic integrity constraints. Related objects not known to be of the constrained type violate the type definition and, thus, make the ontology instance inconsistent with respect to the ontology scheme. Instead, in the case of the mutually exclusive of implied type, related objects are inferred to be of the implied type by a reasoning engine. The additional constraints has min cardinality and has max cardinality restrict the number of objects related to the current one. Last, properties can be attached to the relationships in case the related objects are not data values but belong to a concept. These properties may express transitivity, symmetry, reflexivity, and inverse objects. The last kind of relationship models concept and role assertions. The is member of fact type asserts membership of objects (instances) of a concept. An instance may be member of more than one concept and gives values for the attributes defined in them fulfilling the above requirements for

relationships. The values for the attributes indicate role assertions and are modeled with ...for... has instance value and ... for ... has data value. In the former case, the role assertion is for an instance, and the latter is for a data value.

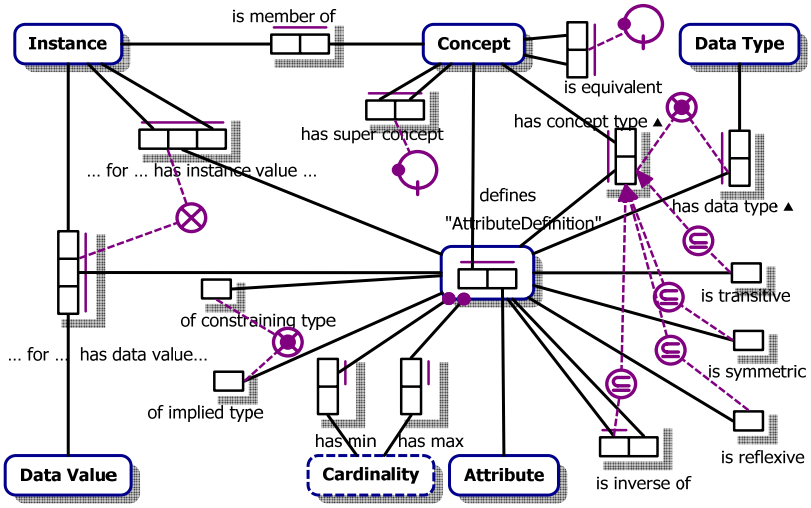

Fig. 3. Concept and Instance modeling with ORM2

Relations and Relation Instances. The semantic relationships that link concepts may be also depicted as relations in WSML and exist outside the boundaries of the concepts definition. The main difference of attribute definitions and relations is that the former are always binary relationships, while relations can have arbitrary arity. An attribute definition links two concepts or a concept and a data type, whereas relations link many concepts and data types in arbitrary order.

The relationships identified for a relation are: (i) ISA relationships used to create hierarchy of relations, (ii) roles played by concepts in the relation, (iii) membership relationships indicating the objects instantiating the relation and fulfilling its roles. With respect to the relationships found in a concept, a relation does not indicate constraints except on the concept types playing roles within it.

In Fig. 4, Relation is the only construct for a relation. The has super relation and is equivalent fact types model inclusion and equality axioms, respectively, and are defined in the same way as the equivalent fact types are defined in a concept. Roles played by some concepts in the relation map to parameter definitions, each one identified by the order in the relation as depicted with RelationHasParameterInPosition. Here too has concept type and has data type are mutually exclusive, and denote the type of the objects playing the corresponding role. Multiple defined types are interpreted as their disjunction. Also the of constraining type and of implied type are defined as for concepts. The last kind of relationship models relation and role assertions. instantiates asserts membership of objects for a relation, called relation instances. A relation instance is for one relation and gives values for the parameters defined in it. The values for the parameters indicate role assertions, modeled with ...for... has instance value and ... for ... has data value.

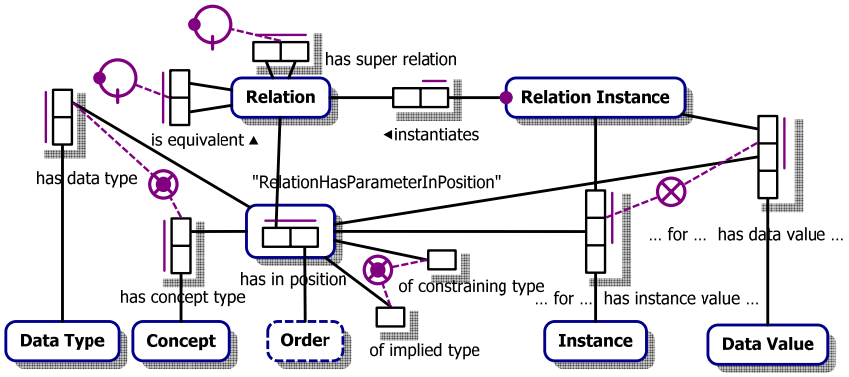

Fig. 4. Relation and Relation Instance modeling with ORM2

Alternatively, relations in WSML with two parameters can be modeled as WSML attribute definitions as well, and relations with more than three parameters as WSML concepts. In the first case, it is good practice to model binary relationships between concepts that are significant for the first related concept as attributes within its definition. At the instance level, the values for these attributes characterize the instance of the concept. For example, in the concept *Human*, gender can be modeled as an attribute *ofGender* and not as a relation between the concepts *Human* and *Gender*, because we expect to indicate its value for every instance of *Human*; later it may be used to partition the population of humans into *Men* and *Women*. As a counter example, the relationship that indicates the spouse of a human is better modeled as a relation *isMarriedTo* between *Human*s because it adds knowledge that does not constrain the definition for *Human*. Extending the arity of the relation *isMarriedTo* for modeling also the wedding date makes less clear the distinction between concepts and relations in WSML. Indeed, the relation that identifies the spouses and the wedding date exists only for this relation and characterizes it; therefore, *isMarriedTo* can be modeled as a concept *Marriage* with three attributes. Such a transformation cannot be avoided if other attribute definitions within concepts relate to the newly created concept *Marriage*. This is exemplified with the case of the concept *Church* in which we want to model the hold out marriages as an attribute that links to the concept *Marriage*.

4.3 Common Elements

WSML defines also some terms that are not first citizens but may be found in the definitions for the already presented WSML elements; the corresponding ORM2 diagrams are in [18] due to space limitations. WSML adopts the International Resource Identifier (IRI) as identification mechanism and it allows anonymous identifiers to indicate elements whose name is unknown or irrelevant. The identifiers follow the Unique Name Assumption (UNA), although some WSML variants allow meta-modeling of objects and hence, the same IRI can be depicted as concept and

instance at the same time. Data types are defined parameterized and are also identified by IRIs. Data values are allowed as values for attributes and parameters. Finally, meta-data may be attached to some of the elements in an ontology specification.

5 Evaluation of the ORM model

We identify two orthogonal dimensions in the evaluation of the produced meta-model for ontologies with ORM2, which are driven by the initial goals that motivated our work. First, we evaluate its capability to model WSML and therefore ORM2's expressivity for being able to represent an ontology language, and, second, its utility within applications for the development of ontologies and of systems that use them.

Beyond the modeling of concepts and relations, also axioms, which are usually depicted textually or formulated in languages like first order logic and description logic, can be easily modeled with ORM2 and its graphical elements. The WSML axioms depicted in our ORM2 model are: inclusion and equality axioms used for creating hierarchy of concepts, relational properties expressing transitivity, symmetry, reflexivity, inverse relations, and cardinality constraints for restricting the number of related concepts reflecting the universal and existential quantifiers. The structure of the axioms is further refined with ORM2 because of the sophisticated features it provides. Ring constraints in Fig.3 and Fig. 4 indicate that inclusion and equality axioms cannot be reflexive, thereby avoiding unnecessary reasoning, and role constraints in Fig.3 between Concept and Data Type indicate that relationship properties characterize relations either between concepts or between data types, but not a concept and a data type. These constructs in ORM2 model part of the specification for an ontology language that cannot be depicted in other formalisms (i.e., ER), or can be represented only partially in UML class diagrams with the additional Object Constraint Language. The advantage of the explicit representation of the axioms is the easy manipulation of them from external tools that are unaware of their implementation language. Thus, the developer focuses on the knowledge she wants to add to the ontology without having to bother herself with the implementation syntax.

The conceptual model of the WSML language for ontologies can be used to augment the documentation provided by the WSML group and by tools for development of WSML ontologies. Regarding the former, it makes explicit the structure of WSML ontologies and resolves unambiguities that have arisen from the natural-language documentation as has been presented in Section 4. In addition, the formal ORM2 model adds clarity to the documentation because of both the graphical elements and the verbalization in pseudo-natural language [18]. ORM has been formalized in first order logic [9] and mapped into $\mathcal{DLR}_{ifd}$ [11]; therefore, the model's satisfiability can be checked on possible errors that may have been created during the modeling of the WSML specification. The representation of the ORM2 diagrams for the WSML specification in first order logic is in [19]. The semantics of the final model remain those given by the WSML group [5].

At the application level, the utility of the conceptual model is twofold. First, it may be used in ontology engineering as well as in ontology-based software development because of the similarities these two fields have. Ontology development tools can

implement the conceptual model presented here by incorporating, or taking as basis, the corresponding relational schema and corresponding DDL scripts for various RDBMSs as well as VB and C# [Halpin, pers. comm.] that are automatically generated with NORMA. An interesting application of the presented ORM2 model is in the implementation of ontology-driven information systems, because the explicit representation of the ontology scheme can be used at design time as well as at runtime. For instance, [17] exemplify the use of the ontology scheme depicted as instances of the appropriate entities in the WebML [20] model both for reasoning at the application level and for providing personalization of data published within Web applications. Second, the main requirement from ontology engineering is the ability to produce a knowledge-based description of the domain. The description must be complete--reflecting all the concepts in the domain and their phenomena--and should not be dependent on the implementation and storage language for ontologies. This raises the need to have a common meta-model for the existing ontology languages. The current conceptual model reflects the expressivity of WSML as ontology language, but it can be easily extended to model also components from other ontology languages such as OWL 1.1 [13]. Thanks to the stability of the fact-oriented modeling mechanism, one can have a smooth, non-disruptive evolution of the ORM model respecting changes in ontology languages, since adding new facts does not require changes of the existing elements. WSML ontologies can have various implementations, where the common model facilitates their exchange and reuse.

6 Conclusions and Future Work

This paper presented a meta-model in ORM2 for representation of ontologies. In particular, we specified how the ORM2 components can be used to express ontological terms and to satisfy ontological modeling requirements that go beyond the modeling of concepts and relations between concepts. The expressivity of ORM2 facilitates the explicit representation of the Web Service Modeling Language family for ontology languages, and may be used to model ontologies independently of their implementation syntax. Such an approach abstracts the semantic knowledge about ontological constructs, axioms, and properties from their implementation in arbitrary formats. Hence, our main contributions consist of: (i) providing the software developer a *formal* WSML specification with the ORM2 conceptual model, which specifies the elements and constraints of the ontology language and its usage, and, (ii), a common conceptual model that can be shared by applications for development of both ontology development tools and ontology-enhanced information systems, and are thereby fully interoperable, At present, we are implementing the current meta-model in WebML [20], which is a high-level language for modeling data-intensive Web applications to create an infrastructure for an RDBMS-based repository for WSML ontologies. We commenced assessing the model for capturing other very expressive ontology languages, such as OWL1.1 [13]. Having such an enriched conceptual model increases its ability to describe a domain; thus, the model becomes powerful and more independent of any standard and new ontology languages. Once a common meta-model for ontologies with ORM2 is established, we will provide

translation mechanisms as bijective functions for mapping ontologies developed with the ORM2 model to their concrete representation in languages like WSML and OWL. Future work will include use of the enhanced model in fields like Web engineering for the design of Semantic Web applications that exploit ontological contents.

References

1. Acciarri, A., Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Palmieri, M., Rosati, R.: QuOnto: Querying ontologies. In Proc. of AAAI 2005, 2005
2. Antezana, E., Tsiporkova, E., Mironov, V., Kuiper, M.: A cell-cycle knowledge integration framework. DILS 2006, LNBI 4075: 19-34.
3. Bechhofer, S., Horrocks, I., Turi, D.: The OWL Instance Store: System description. In Proceedings of CADE-20, LNCS. Springer Verlag, 2005.
4. De Bruijn, J., Heymans, S.: A semantic framework for language layering in WSML. RR2007, LNCS 4524, 103-117.
5. De Bruijn, J., Heymans, S.: WSML Ontology Semantics. WSML Working Draft D28.3v01. Available from http://www.wsmo.org/TR/d28/d28.3/v0.1/20061218/.
6. De Bruijn, J., Lausen, H., Krummenacher, R., Polleres, A., Predoiu, L., Kifer, M., Fensel, D.: The Web Service Modeling Language WSML. WSML Working Draft D16.1v0.21. Available from http://www.wsmo.org/TR/d16/d16.1/v0.21/.
7. Gennari, J.H., Musen, M.M., Fergerson, R.W., Grosso, W.E., Crubézy, M., Eriksson, H., Noy, N.F., Tu, S.W.: The Evolution of Protégé: An Environment for Knowledge-Based Systems Development. Int. J. of Human-Computer Studies, 2003, 58(1):89-123
8. Gene Ontology, http://www.geneontology.org/
9. Halpin, T.A.: A logical analysis of information systems: static aspects of the data-oriented perspective. PhD Thesis, University of Queensland, Australia. 1989.
10. Keet, C.M.: Granular information retrieval from the Gene Ontology and from the Foundational Model of Anatomy with OQAFMA. KRDB Research Centre Technical Report KRDB06-1, Free University of Bozen-Bolzano, 6 April 2006.
11. Keet, C.M.: Prospects for and issues with mapping the Object-Role Modeling language into DLRifd. In Proceedings of DL 2007. CEUR-WS Vol-250, 331-338.
12. Neumont ORM Architect, http://sourceforge.net/projects/orm/.
13. OWL 1.1, (Editor's draft of 23-5-2007). http://www.webont.org/owl/1.1/.
14. Poggi, A., Ruzzi, M.: Ontology-based data access with MASTRO. OWL: Experience and Directions (OWLED 2007). Innsbruck, Austria, 6-7 June, 2007
15. Protégé, http://protege.stanford.edu/.
16. Rosse, C., Mejino, J.L.V.: A reference ontology for biomedical informatics: the foundational model of anatomy. Journal of Biomedical Informatics, 2003, 36: 478-500.
17. Tziviskou, C., Brambilla, M.: Semantic Personalization of Web Portal Contents. Poster at ACM WWW 2007, Banff Canada, May 2007.
18. Tziviskou, C., Keet, C.M.: ORM2 diagrams for WSML specification. Available from http://www.elet.polimi.it/upload/tzivisko/wsml-orm2-norma.pdf
19. Tziviskou, C., Keet, C.M.: Translation of ORM2 diagrams for WSML specification to First Order Logic, http://www.elet.polimi.it/upload/tzivisko/wsml-orm-fol.htm
20. WebML, http://www.webml.org.
21. WSML, http://www.wsmo.org/wsml/.